Technical white paper

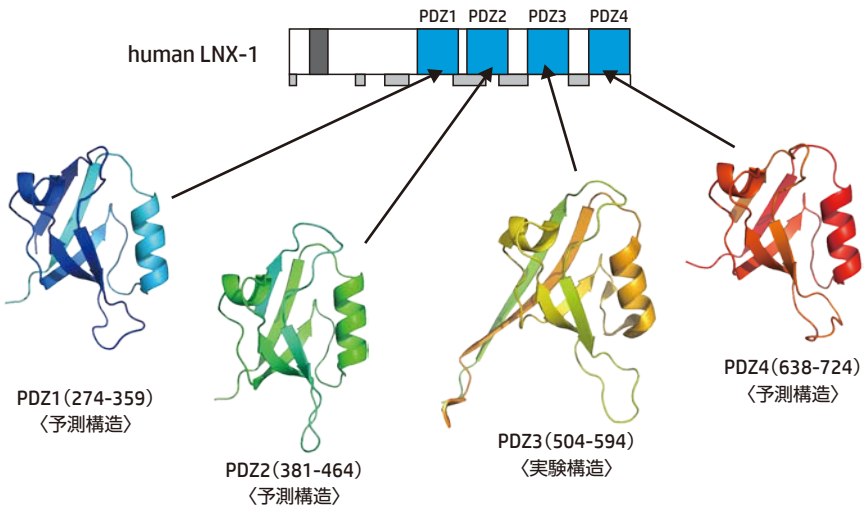
# HP ProLiant DL980 G7を使ったドメインドッキングの高速化

人の生命活動に重要な役割を果たすタンパク質は、何らかの機能を発揮しようとする際に、タンパク質や核酸などの分子が緩やかに結合した巨大な超分子が生体内で形成される。その立体構造を理解することは、より効果的に作用し、副作用も少ない医薬品を開発するために大きな意味を持つ。しかし、複数の分子が複雑に組み合わさった超分子の場合、従来の実験をベースとしたアプローチで立体構造を正確に把握することが困難だった。こうした課題の解決に向け、滋賀県にある長浜バイオ大学では、バイオインフォマティックス技術に基づきコンピューター上で超分子モデリングを実行するための基盤技術を確立する研究が進んでいる。この研究の重要な構成要素の一つであるドメインドッキングに、インテル® Xeon®プロセッサーE7ファミリー8基（80コア）を搭載したHP ProLiant DL980 G7を適用。Open MPI版ZDOCK、Pythonのmultiprocessingパッケージを利用し、処理の並列化を行うことでドッキング処理にかかる時間を大幅に短縮できるメドが立ちつつある。このホワイトペーパーでは、研究を担当する同大学バイオサイエンス学部の塩生真史講師が行ったベンチマーク結果をご紹介する。

## 1. ドメインドッキングについて

超分子は複数のタンパク質や核酸などで構成されている。この中のタンパク質に注目すると、特定の機能を持つ「ドメイン」と呼ばれる単位が複数連結しているタンパク質が多い（図1）。複数のドメインが連結したタンパク質が立体構造を形成する際には、それぞれのドメイン表面の立体的な凹凸や電気的特性などによって安定した状態をとると考えられている。

**図1.** 複数のドメインが連結しているタンパク質の例（「human LNX-1」の場合）



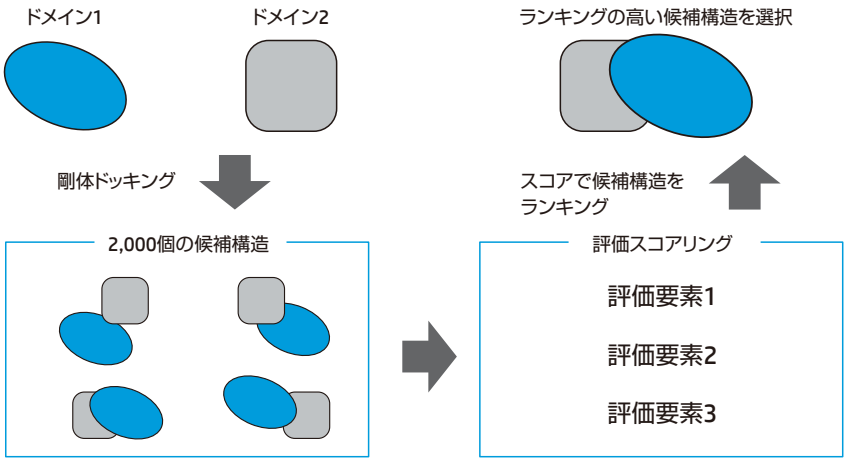

intel inside Xeon
インテル® Xeon® プロセッサー
E7 ファミリー

## 2. ドメインドッキングによる2ドメインタンパク質の立体構造予測フロー

ドメインドッキングによって複数のドメインが連結したタンパク質の立体構造を予測する際に、まず最初のステップとなるのは、連結した2つのドメインの立体配置を正確に予測することである。塩生講師が考案したドメインドッキングの処理フローは以下のようなものだ。まず、2つのドメインの一方を固定し、もう一方のドメインを並進・回転させながらコンピューター上で剛体ドッキングを行い、候補構造を2,000個ほど生成する。得られた候補構造を、ドメイン同士の接触に使われやすいアミノ酸かどうか、といった3つの指標により評価してスコア化し、これらを足し合わせてランキングを作成。上位10位以内にランキングされた候補構造を、現実の立体構造の候補構造として最終的に抽出する(図2)。

以上のような処理フローを、実験から立体構造の分かっている2ドメインタンパク質で検証した結果、従来の方法と比較して、最終的な候補構造の中に現実のものと極めて類似した立体構造が含まれる割合が高く、2ドメインタンパク質の立体構造を高い精度で予測できることが確認できた。

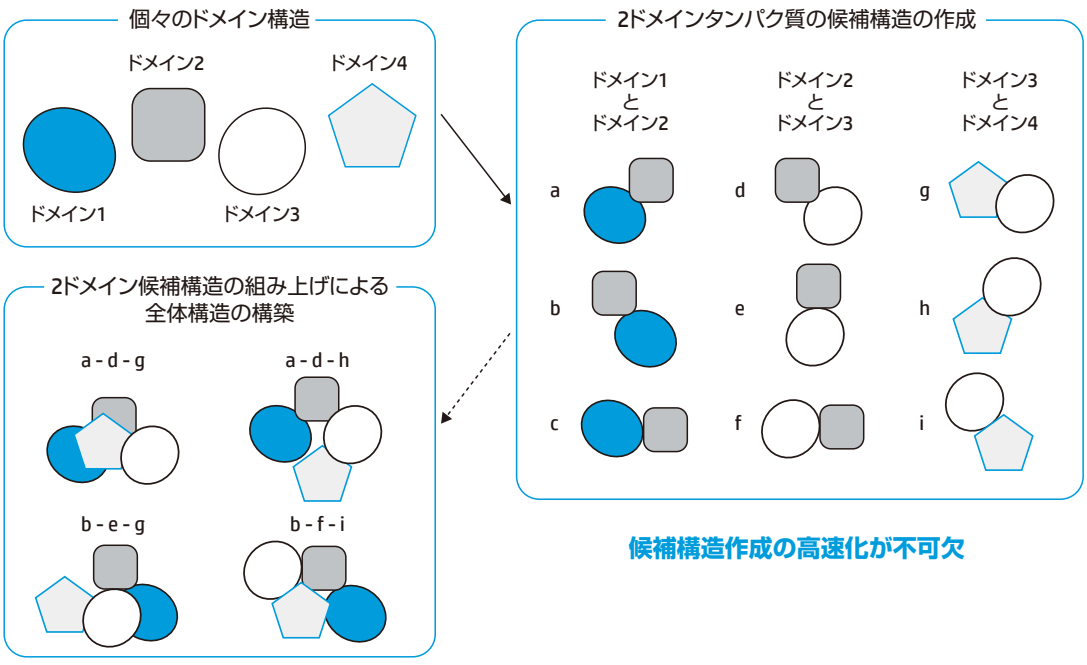
**図2.** 2ドメインタンパク質の構造予測フロー




## 3. 最終候補構造の絞り込みを短時間で行えることが鍵

塩生講師の長期的な研究目標は、2ドメインタンパク質での成果を拡張し、3ドメイン、そして4ドメインと連結するドメインの数を増やしていった時にも、より正確な立体構造を予測できるような処理フローを確立することだ。当面は4ドメインまでの拡張を目指している(図3)。連結するドメインの数を増やす研究を効率良く進めていくには、2ドメインタンパク質の候補構造をいかに短時間で生成し、最終候補構造を抽出できるかが大きな鍵となる。

**図3.** 構造予測の方針



## 4. HP ProLiant DL980 G7、Open MPI版ZDOCK、Pythonで環境を構築

2ドメインタンパク質の最終候補構造の抽出時間を短縮するため、塩生講師が採用したアイディアは並列処理である。抽出までの第1ステップとなる2ドメインタンパク質の候補構造生成のためのドッキング処理、そして第2ステップの評価スコアリング処理のそれぞれをマルチコアで並列化できれば、大きな時間短縮が期待できる。

そのために選んだハードウェア環境が、インテル® Xeon®プロセッサーE7ファミリー8基（80コア）を搭載したHP ProLiant DL980 G7である。ソフトウェア環境としては、OSにCentOS。ドッキング処理には、この分野で広く利用されており、並列処理機能を容易に追加できるOpen MPIに対応したZDOCKを選択。評価スコアリング処理は、スクリプト言語Pythonのmultiprocessingパッケージを使用した（図4）。シングルプロセスをマルチプロセス化するにはアルゴリズムやプログラミングなどに関する高度な専門知識が必要になるものと考えられがちだ。しかし塩生講師は、特別に難しいことはしていない、という。プログラミングの基礎的な知識があれば「pythonの勉強に1日当てた程度」（塩生氏）で、容易に今回のベンチマーク環境を作り上げることができた、とのことだ。

**図4.** ベンチマークのためのシステム環境

**ハードウェア:HP ProLiant DL980 G7**
搭載プロセッサー:インテル® Xeon® プロセッサー E7-4870 2.4GHz 8基／80コア
搭載メモリー:1TB
搭載記憶領域:900GB SAS HDD×8台　合計7.2TB

**OS:CentOS**

**アプリケーション:Open MPI版ZDOCK**
**Pythonのmultiprocessingパッケージ**

**図5.** 自作したプログラムの一部

```
    def RunZDOCK(self):
        mark_sur = self.zdock_dir + '/mark_sur'
        zdock = self.zdock_dir + '/zdock'
        if os.path.exists(self.work_dir + '/uniCHARMM') == False:
            shutil.copy(self.zdock_dir + '/uniCHARMM', self.work_dir)
        pdbFiles = []
        dockFiles = []
        for tag in (1, 2):
            domainFile = self.work_dir + '/domain' + str(tag) + '.pdb'
            pdbFile = self.work_dir + '/pdb' + str(tag) + '.pdb'
            dockFile = self.work_dir + '/dock' + str(tag) + '.out'
            pdbFiles.append(pdbFile)
            dockFiles.append(dockFile)
            subprocess.check_call([mark_sur, domainFile, pdbFile])

        cmd = []
        cmd.append([zdock, '-o', dockFiles[0], '-N', str(self.pose_num), '-S', '1', '-F', '-R', pdbFiles[0], '-L', pdbFiles[1]])
        cmd.append([zdock, '-o', dockFiles[1], '-N', str(self.pose_num), '-S', '2', '-F', '-R', pdbFiles[1], '-L', pdbFiles[0]])
        processList = []
        for tag in (1, 2):
            if self.coreNum == 1:
                self.ExecZDOCK(tag, cmd[tag-1])
            else:
                process = multiprocessing.Process(target = self.ExecZDOCK, args = (tag, cmd[tag-1], ))
                processList.append(process)
                process.start()
        for process in processList:
            process.join()

        self.ModifyPDB()

    def ExecZDOCK(self, tag, cmd):
        if self.processNum == 1:
            subprocess.call(cmd)
        else:
            mpi = ['/usr/local/bin/mpiexec', '-n', str(self.processNum)]
            subprocess.check_call(mpi + cmd)
        self.ModifyZDOCK(tag)

```

## 5. 80コアを使えば、処理時間はシングルコアの1/33以下に

図4で示したシステム環境を利用し、HP ProLiant DL980 G7の使用コア数を増やしながら、2ドメインタンパク質の最終候補構造を抽出する処理フローにかかった時間を計測した結果が図6である。77個の2ドメインタンパク質に対して処理フローを繰り返し、1タンパク質あたりにかかった時間の中央値を示している。図7は、使用コア数ごとの効率化度合を示したもので、シングルコアでの処理時間を1としている。

図6から、シングルコアでは1,187秒（約20分）かかっていた処理が、80コアで並列化することにより、わずか36秒に大幅に短縮できたことが分かる。「既存のハードウェアとソフトウェアをほぼそのまま使うことで、ここまで高速化できたことには正直驚いている」と塩生講師も喜びを隠さない。処理時間が劇的に速くなることで、今後の取り組みにも大きな貢献が期待できる。

**図6.** 処理時間の中央値

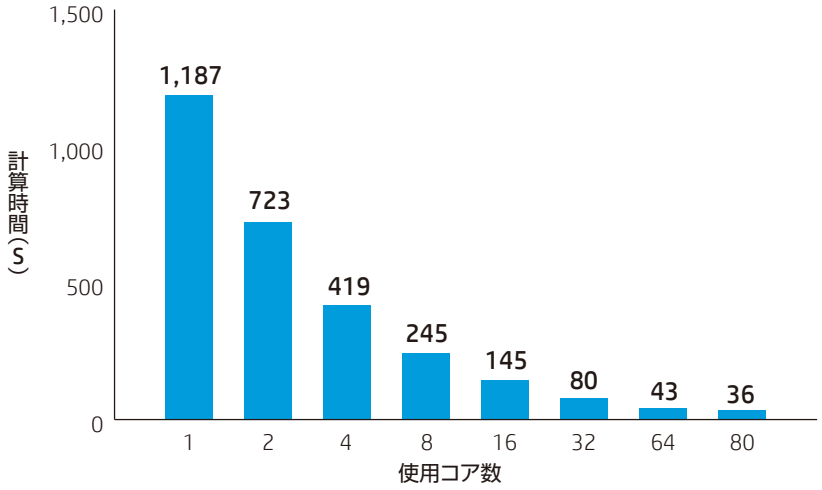
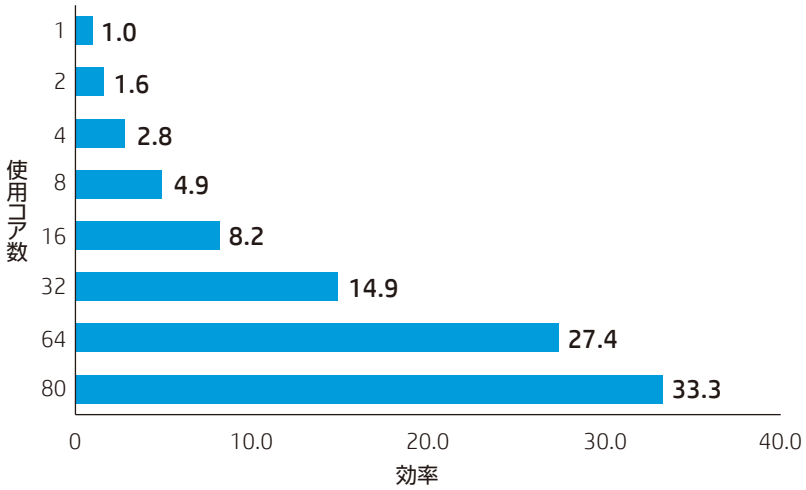



**図7.** 並列化による処理効率の違い




安全に関するご注意　ご使用の際は、商品に添付の取扱説明書をよくお読みの上、正しくお使いください。水、湿気、油煙等の多い場所に設置しないでください。火災、故障、感電などの原因となることがあります。

お問い合わせはカスタマー・インフォメーションセンターへ
03-5749-8328 月～金 9:00～19:00　土 10:00～17:00（日、祝祭日、年末年始および5/1を除く）
機器のお見積もりについては、代理店、または弊社営業にご相談ください。
HP ProLiantに関する情報は http://www.hp.com/jp/proliant

Intel、インテル、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Xeon、Xeon Insideは、アメリカ合衆国および / またはその他の国におけるIntel Corporationの商標です。
記載されている会社名および商品名は、各社の商標または登録商標です。
記載事項は2013年3月現在のものです。
本カタログに記載されている情報は取材時におけるものであり、閲覧される時点で変更されている可能性があります。あらかじめご了承ください。
© Copyright 2013 Hewlett-Packard Development Company,L.P.

本カタログは、環境に配慮した用紙と植物性大豆油インキを使用しています。 PRINTED WITH SOY INK

日本ヒューレット・パッカード株式会社
〒136-8711 東京都江東区大島2-2-1